

Axiomatic Systems for the Bisimilarity on Finite Fair Ambient Processes

Han Zhu

Basics Lab, Department of Computer Science
Shanghai Jiao Tong University, Shanghai, China
Email: zhu-h@sjtu.edu.cn

Abstract—In this paper, we study an axiom system for the bisimilarity on finite Fair Ambient processes. In order to obtain normal forms of finite processes, we extended the syntax of Fair Ambient to put the nested ambient structure into prefixes. Upon our axiom system, if two finite Fair Ambient processes are equivalent can be effectively checked.

Index Terms—Fair Ambient, Axiomatization, Bisimilarity

I. INTRODUCTION

Process calculi are mathematical models for describing and analyzing properties of distributed concurrent systems. As a result of it, concurrent processes exhibit more complicated behavior than traditional sequential processes. Bisimulation relations have attracted a lot of attention paid to giving process calculi a reasonable equivalence relation. These equivalence relations are subject to the examinations of a process's internal states.

The Calculus of Mobile Ambients, MA for short, is introduced by Cardelli and Gordon [1], [2], [3] as a model that provides a uniform account for both mobile computation and mobile computing [4]. The apparatus introduced by MA is ambient, a piece of program residing in a specified location. A piece of resource may move from one ambient to another. Several variants of MA, like Safe Ambients [5], Safe Ambients with Passwords [6], Controlled Ambients [7] and ROAM (The Calculus of Robust Ambients) [8] to name a few, have been proposed to enhance the control power of MA. The Calculus of Fair Ambient [9], or FA, has the most strict rules about when ambients may interact. It achieves that by sticking to the following principles:

- 1) First Class Citizenship: All actions are interactions between ambients.
- 2) Authorization: The two ambients involved in an interaction must obtain the authorizations from each other.
- 3) Authentication: The two ambients involved in an interaction must know each other's identities.

It is proved in [9] that the π -calculus [10], [11] is a sub-calculus of FA.

The Turing completeness of ambient calculi defeats any attempt to obtain a sound and complete axiom system. But if we restrict our consideration to finite processes, something positive can be achieved. Now we are interesting in axiomatizing the bisimilarity relation \approx on finite Fair Ambient processes.

Fair Ambient uses an interleaving semantics of concurrency, just as other members of process calculi family do. By the notion of “+”, the nondeterministic choice we will introduce, along with the well-known expansion law, any finite process can be rewritten into a normal form, which in turn has the same behaviors as the original one. That is to say, they are equivalent with respect to the specific equivalence relation we are axiomatizing. Then an equational theory on normal forms is given. This is the common approach towards axiomatizing an equivalence relation.

We adopt the common approach, but also are exposed to some issues specific to FA. Besides the need to introduce an additional operator “+”, which is not defined in the original FA, we should tackle higher order communications, parallel operators in normal form, etc.

Fair Ambient, like its predecessor Mobile Ambient, is inherently higher order, where processes can move in or move out from one position (ambient) to another position (ambient). Higher order communications, i.e. processes can be communicated, are adopted to achieve these movement capabilities.

To avoid introducing higher order labels in the operational semantics, we use contexts and concretions. They are all intermediate forms of an ambient in an interaction, not final forms after an interaction. But in axiomatizing, they are all required to construct the normal form of an ambient. So we borrow them from the operational semantics of FA (labeled transition system). Due to the existence of contexts and concretions, the parallel operators can not be fully eliminated by the expansion law in the normal form.

Ambient capabilities should also be extended to contain action labels from labeled transition system, and the non-deterministic choice will be defined as a guarded choice among them to eliminate parallel composition operators to some extent. These considerations lead to an extended fair ambient calculi, in which each finite fair ambient will have

This work is supported by the National 973 Project (2003CB317005) and the National Nature Science Foundation of China (60573002, 60703033).

a corresponding strong bisimilar normal form extended fair ambient.

This paper is structured as follows. Section II provides some background knowledge of the Calculus of Fair Ambient. Definitions of extended Fair Ambient and how to translate any finite processes into normal forms are examined in Section III. Section IV goes through the sound and complete axiom system for strong bisimilarity. Some examples of checking if two processes are bisimilar are illustrated in Section V. Finally, Section VI concludes this paper.

II. THE CALCULUS OF FAIR AMBIENT

The semantics of FA is defined purely in terms of a labeled transition system, whose rules are carefully chosen to adhere to the three principles advocated above.

The syntax of processes is defined by the following grammar:

$$P ::= \mathbf{0} \mid \kappa.P \mid P \mid P \mid a[P] \mid (a)P \mid !P$$

where κ is moving capability:

$$\kappa ::= \text{in } a \mid \overline{\text{in}} a \mid \text{out } a \mid \overline{\text{out}} a \mid \text{open } a \mid \overline{\text{open}} a$$

Interactions between ambients are higher order, and FA uses concretion and contexts to avoid higher order labels. Concretions have the form of $\langle P \rangle P'$ where $\langle P \rangle$ is the moving part and P' the remaining part. Contexts have the form of $C[_]$, where exactly one hole $_$ is in it.

Action labels of operational semantics are defined as:

$$\begin{aligned} \lambda ::= & \kappa \mid \tau \\ & | a[\text{in } b] \quad a \text{ moves into } b \\ & | a[\overline{\text{in}} b] \quad a \text{ imports } b \\ & | [\text{out } b] \quad \text{moves out to reach } b \\ & | a[[\text{out } b]] \quad \text{an ambient moves out of } a \text{ to reach } b \\ & | a[\overline{\text{out}} b] \quad a \text{ extracts an ambient out of } b \\ & | a[\text{open } b] \quad a \text{ opens up } b \\ & | a[\overline{\text{open}} b] \quad a \text{ is opened up by } b \end{aligned}$$

Labeled transition system are divided into three groups of rules.

1) Structural Rule:

$$\frac{}{\kappa.P \xrightarrow{\kappa} P} \quad \frac{P \xrightarrow{\lambda} U}{P \mid Q \xrightarrow{\lambda} U \mid Q} \quad \frac{P \xrightarrow{\tau} P'}{a[P] \xrightarrow{\tau} a[P']} \\ \frac{P \xrightarrow{\lambda} U \quad n \text{ is not in } \lambda}{(n)P \xrightarrow{\lambda} (n)U} \quad \frac{P \xrightarrow{\lambda} U}{!P \xrightarrow{\lambda} U}$$

2) Ambient Rule:

$$\frac{P \xrightarrow{\text{in } b} P'}{a[P] \xrightarrow{a[\text{in } b]} \langle a[P'] \rangle \mathbf{0}} \quad \frac{P \xrightarrow{\overline{\text{in}} b} P'}{a[P] \xrightarrow{a[\overline{\text{in}} b]} a[_ \mid P']}$$

$$\frac{P \xrightarrow{\text{out } b} P'}{a[P] \xrightarrow{a[\text{out } b]} \langle a[P'] \rangle \mathbf{0}} \quad \frac{P \xrightarrow{[\text{out } b]} (\tilde{m}) \langle A \rangle P'}{a[P] \xrightarrow{a[[\text{out } b]]} (\tilde{m}) \langle A \mid a[P'] \rangle}$$

$$\frac{P \xrightarrow{\overline{\text{out}} b} P'}{a[P] \xrightarrow{a[\overline{\text{out}} b]} a[P']} \quad \frac{P \xrightarrow{\text{open } b} P'}{a[P] \xrightarrow{a[\text{open } b]} a[_ \mid P']}$$

$$\frac{P \xrightarrow{\overline{\text{open}} b} P'}{a[P] \xrightarrow{a[\overline{\text{open}} b]} \langle a[P'] \rangle \mathbf{0}}$$

3) Interaction Rule:

$$\frac{P \xrightarrow{b[\text{in } a]} (\tilde{m}) \langle B \rangle P' \quad Q \xrightarrow{a[\overline{\text{in}} b]} C[_]}{P \mid Q \xrightarrow{\tau} (\tilde{m}) \langle P' \mid C[B] \rangle}$$

$$\frac{P \xrightarrow{b[[\text{out } a]]} P' \quad Q \xrightarrow{a[\overline{\text{out}} b]} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{a[\text{open } b]} C[_] \quad Q \xrightarrow{b[\overline{\text{open}} a]} (\tilde{m}) \langle Q'' \rangle Q'}{P \mid Q \xrightarrow{\tau} (\tilde{m}) \langle C[Q''] \mid Q' \rangle}$$

Some examples show how an ambient system evolves.

Example 1 (Renaming).

$$b[\text{open } a] \mid a[\overline{\text{open}} b] \mid P \xrightarrow{\tau} b[P]$$

Example 2 (Nondeterministic Choice). Let $d[\overline{\text{open}} a] + d[\overline{\text{open}} b]$ be

$$(c)(c[\overline{\text{open}} d.\overline{\text{open}} a] \mid c[\overline{\text{open}} d.\overline{\text{open}} b] \mid d[\text{open } c])$$

then clearly

$$d[\overline{\text{open}} a] + d[\overline{\text{open}} b] \xrightarrow{\tau} \sim d[\overline{\text{open}} a] \\ d[\overline{\text{open}} a] + d[\overline{\text{open}} b] \xrightarrow{\tau} \sim d[\overline{\text{open}} b]$$

Thus the nondeterministic choice $a[P] + b[Q]$ is then defined by

$$(d)((d[\overline{\text{open}} a] + d[\overline{\text{open}} b]) \mid a[\text{open } d.P] \mid b[\text{open } d.Q])$$

Fair Ambient uses the standard bisimulation method as its equivalence relation.

Definition 1. A symmetric relation \mathcal{R} on processes is a bisimulation if the followings hold whenever $P \mathcal{R} Q$:

- 1) If $P \xrightarrow{\tau} P'$ then $Q \Rightarrow Q' \mathcal{R} P'$ for some Q' ;
- 2) If $P \xrightarrow{\kappa} P'$ then $Q \xrightarrow{\kappa} Q' \mathcal{R} P'$ for some Q' ;
- 3) If $P \xrightarrow{\lambda} P'$ and $\lambda \in \{a[\overline{\text{out}} b], a[[\text{out } b]] \mid a, b \in \mathcal{N}\}$ then $Q \xrightarrow{\lambda} Q' \mathcal{R} P'$ for some Q' ;
- 4) If $P \xrightarrow{[\text{out } a]} (\tilde{m}) \langle A \rangle P'$ then for every name d and every process O satisfying $\{\tilde{m}\} \cap fn(O) = \emptyset$ there exist some \tilde{n}, B, Q', Q'' such that $Q \Rightarrow [\text{out } a] (\tilde{n}) \langle B \rangle Q'$ and $(\tilde{n}) \langle B \mid d[Q' \mid O] \rangle \Rightarrow Q'' \mathcal{R} (\tilde{m}) \langle A \mid d[P' \mid O] \rangle$;
- 5) If $P \xrightarrow{a[\text{in } b]} (\tilde{m}) \langle A \rangle P'$ then for every O satisfying $\{\tilde{m}\} \cap fn(O) = \emptyset$ some \tilde{n}, B, Q', Q'' exist such that

- $Q \Rightarrow^{a[\text{in } b]} (\tilde{n})(\langle B \rangle Q')$ and $(\tilde{n})(Q' | b[B | O]) \Rightarrow Q'' \mathcal{R} (\tilde{m})(P' | b[A | O]);$
- 6) If $P \xrightarrow{a[\text{open } b]} (\tilde{m})(\langle M \rangle P')$ then for every O satisfying $\{\tilde{m}\} \cap fn(O) = \emptyset$ there exist some \tilde{n}, N, Q', Q'' such that $Q \Rightarrow^{a[\text{open } b]} (\tilde{n})(\langle N \rangle Q')$ and $(\tilde{n})(Q' | b[N | O]) \Rightarrow Q'' \mathcal{R} (\tilde{m})(P' | b[M | O]);$
- 7) If $P \xrightarrow{a[\text{in } b]} C[_]$ then for every process N satisfying $fn(N) \cap bn(C[_]) = \emptyset$ there exist some $C'[_], Q'$ such that $Q \Rightarrow^{a[\text{in } b]} C'[_]$ and $C'[b[N]] \Rightarrow Q' \mathcal{R} C[b[N]];$
- 8) If $P \xrightarrow{a[\text{open } b]} C[_]$ then for every process N satisfying $fn(N) \cap bn(C[_]) = \emptyset$ there exist some $C'[_], Q'$ such that $Q \Rightarrow^{a[\text{open } b]} C'[_]$ and $C'[N] \Rightarrow Q' \mathcal{R} C'[N].$

The bisimilarity \approx is the largest bisimulation.

An important result of this bisimilarity is the local observability.

Theorem 1. For FA processes P and Q , it holds that $P \approx Q$ if and only if $C[P] \approx C[Q]$ for every ambient context $C[_].$

III. EXTENDED FAIR AMBIENT AND NORMAL FORM

Definition 2. The syntax of extended fair ambient is defined by the following grammar (a and b range over the set of names \mathcal{N}):

$$P ::= K.G \mid P + P \mid \mathbf{0}$$

$$G ::= P \mid \langle P \rangle \mid P \mid a[_ | P] \mid P \mid (a)G$$

where K is the capabilities defined as:

$$K ::= \tau \mid \text{in } a \mid \overline{\text{in}} a \mid \text{out } a \mid \overline{\text{out}} a \mid \text{open } a \mid \overline{\text{open}} a$$

$$\mid a[\text{in } b] \mid a[\overline{\text{in}} b] \mid [\text{out } b] \mid a[[\text{out } b]]$$

$$\mid a[\overline{\text{out}} b] \mid a[\text{open } b] \mid a[\overline{\text{open}} b]$$

There are three pairs of complementary capabilities:

- 1) $a[\text{in } b]$ and $b[\overline{\text{in}} a];$
- 2) $a[[\text{out } b]]$ and $b[\overline{\text{out}} a];$
- 3) $a[\text{open } b]$ and $b[\overline{\text{open}} a].$

Additional rules of the operational semantics of extended fair ambient is:

$$\frac{}{K.G \xrightarrow{K} G} \quad \frac{P_i \xrightarrow{K} P'_i}{P_1 + P_2 \xrightarrow{K} P'_i} \text{ for } i \in \{1, 2\}$$

Bisimilarity of extended fair ambient is the bisimilarity plus the following two special cases:

Definition 3. The special cases of concretions and contexts:

- $(\tilde{y})(\langle P_1 \rangle | P_2) \sim (\tilde{z})(\langle Q_1 \rangle | Q_2)$ iff

$$\forall a, R_1, R_2: (\tilde{y})(a[R_1 | P_1] | R_2 | P_2) \sim (\tilde{z})(a[R_1 | Q_1] | R_2 | Q_2)$$
- $(\tilde{y})(a[_ | P_1] | P_2) \sim (\tilde{z})(a[_ | Q_1] | Q_2)$ iff

$$\forall R_1, R_2: (\tilde{y})(a[R_1 | P_1] | R_2 | P_2) \sim (\tilde{z})(a[R_1 | Q_1] | R_2 | Q_2)$$

Let us look at some examples to get a picture of how to rewrite any finite fair ambient into its corresponding normal form extended fair ambient. In this section, the symbol \mapsto represents rewriting.

Our aim is to construct a strong bisimilar normal form extended fair ambient. Why we choose strong bisimilarity is that it is a much finer equivalence relation, implying many other equivalence relations, e.g. testing equivalence, weak bisimilarity (the one we are axiomatizing). It is reusable when axiomatizing these equivalence relations.

The following examples illustrate the uses of non-deterministic choices, concretions and contexts in normal forms.

Example 3.

$$\text{in } a \mid \text{in } b \mapsto \text{in } a.\text{in } b + \text{in } b.\text{in } a$$

$$a[\text{in } b.P] \mapsto a[\text{in } b].\langle a[P] \rangle$$

$$a[\overline{\text{in}} b.P] \mapsto a[\overline{\text{in}} b].a[_ | P]$$

The next couple of examples gives a taste of what a normal form would be.

Example 4 (Concretions).

$$b[\text{in } a] \mid c[\text{in } a]$$

$$\mapsto b[\text{in } a].\langle b[_] \rangle \mid c[\text{in } a].\langle c[_] \rangle$$

$$\mapsto b[\text{in } a].(\langle b[_] \rangle \mid c[\text{in } a].\langle c[_] \rangle)$$

$$+ c[\text{in } a].(b[\text{in } a].\langle b[_] \rangle \mid \langle c[_] \rangle)$$

Example 5 (Contexts).

$$b[\overline{\text{in}} a] \mid d[\overline{\text{in}} a]$$

$$\mapsto b[\overline{\text{in}} a] \mid d[\overline{\text{in}} a].d[_]$$

$$\mapsto b[\overline{\text{in}} a].d[\overline{\text{in}} a].d[_] + d[\overline{\text{in}} a].(\overline{\text{in}} a \mid d[_])$$

$$\mapsto b[\overline{\text{in}} a].b[_ \mid d[\overline{\text{in}} a].d[_] + bd[\overline{\text{in}} a].b[\overline{\text{in}} a \mid d[_]]$$

$$\mapsto b[\overline{\text{in}} a].b[_ \mid d[\overline{\text{in}} a].d[_]]$$

$$b[\overline{\text{in}} a] \mid c[\overline{\text{in}} a]$$

$$\mapsto b[\overline{\text{in}} a].b[_ \mid c[\overline{\text{in}} a].c[_]]$$

$$\mapsto b[\overline{\text{in}} a].(b[_ \mid c[\overline{\text{in}} a].c[_] + c[\overline{\text{in}} a].(b[\overline{\text{in}} a].b[_ \mid c[_]])$$

From the above examples, we have a sense that a normal form should be a summation of $K.(\tilde{x})P$, $K.(\tilde{x})(\langle P' \rangle | P)$ and $K.(\tilde{x})(a[_ | P'] | P)$. The following definition and theorem formalize this intuition.

Definition 4. P is a normal form, or is in normal form, if P is in the form of

$$\sum_{i=1}^l K_i.(\tilde{x})P_i + \sum_{j=1}^m K_j.(\tilde{y})(\langle P'_j \rangle | P_j)$$

$$+ \sum_{k=1}^n K_k.(\tilde{z})(a_k[_ | P'_k] | P_k)$$

where $P_i, P_j, P'_j, P_k, P'_k$ are also in normal form.

Note that the nil process $\mathbf{0}$ is in normal form, since we can get it by choosing $l = m = n = 0$.

The depth of a process measures the number of nested prefixes in its syntactic representation. Given finite processes, it usually equals to the maximal actions a finite process could do. Its structural definition goes as follows:

- 1) $d(\mathbf{0}) = 0$;
- 2) $d(K.P) = d(P) + 1$;
- 3) $d(P_1 | P_2) = d(P_1) + d(P_2)$;
- 4) $d(a[P]) = d(P)$;
- 5) $d((a)P) = d(P)$;
- 6) $d(P_1 + P_2) = \max\{d(P_1), d(P_2)\}$;
- 7) $d(\langle P \rangle) = d(P)$;
- 8) $d(_) = 0$.

Theorem 2. *There exists a rewriting procedure, which can rewrite any finite fair ambient P to a strong bisimilar normal form extended fair ambient P' with equal or smaller depth.*

Proof: The rewriting procedure runs recursively in a structural way. We will show that in the same way the procedure meets our requirements. In other words, we will demonstrate that we obtain a strong bisimilar process with equal or smaller depth in each recursive call no matter which operator the procedure is processing.

- The cases for $\mathbf{0}$, $K.P$ and $P_1 + P_2$ are obvious since they are already in normal form if P , P_1 and P_2 are.
- We also preserve the form of $\langle P \rangle$ and $_$ when P is already in normal form.
- The axiom

$$(a)(P + Q) = (a)P + (a)Q$$

which will be listed in the table of axioms, can be used for restriction $(a)P$, in case of $P \equiv \sum K_i.P_i$ is in normal form. Note that axioms

$$\begin{cases} (a)K.P = \mathbf{0} & \text{if } a \in n(K) \\ (a)K.P = K.(a)P & \text{if } a \notin n(K) \end{cases}$$

are useful to push restriction inwards and eliminate those actions invisible from outside. In the above, $\lambda K.n(K)$ is a function mapping capability K to the subjective names in it:

$$n(K) = \begin{cases} \emptyset & \text{if } K \equiv \tau \\ \{a\} & \text{if } K \in A \\ \{a, b\} & \text{if } K \in B \end{cases}$$

where

$$A = \{\text{in } a, \overline{\text{in}} a, \text{out } a, \overline{\text{out}} a, \text{open } a, \overline{\text{open}} a, [\text{out } a]\}$$

and

$$B = \{a[\text{in } b], a[\overline{\text{in}} b], a[[\text{out } b]], a[\overline{\text{out}} b], a[\text{open } b], a[\overline{\text{open}} b]\}$$

- Rewriting $a[P]$ into its normal form plays a crucial part in our procedure. If $P \equiv \sum K_i.P_i$ is already in normal form, rewriting takes two steps. The first step is

$$a[\sum K_i.P_i] \mapsto \sum a[K_i.P_i]$$

The second step rewrites each $a[K_i.P_i]$ according to the form of K_i :

- 1) If $K_i \equiv \tau$, then $a[\tau.P_i] \mapsto \tau.a[P_i]$;
- 2) If $K_i \equiv \text{in } b$, then $a[\text{in } b.P_i] \mapsto a[\text{in } b].\langle a[P_i] \rangle$;
- 3) If $K_i \equiv \overline{\text{in}} b$, then $a[\overline{\text{in}} b.P_i] \mapsto a[\overline{\text{in}} b].a[_ | P_i]$;
- 4) If $K_i \equiv \text{out } b$, then $a[\text{out } b.P_i] \mapsto [\text{out } b].\langle a[P_i] \rangle$;
- 5) If $K_i \equiv [\text{out } b]$, then $a[[\text{out } b].(\tilde{x})(\langle P'_i \rangle | P''_i)] \mapsto a[[\text{out } b].(\tilde{x})(P'_i | a[P''_i])]$;
- 6) If $K_i \equiv \overline{\text{out}} b$, then $a[\overline{\text{out}} b.P_i] \mapsto a[\overline{\text{out}} b].a[P_i]$;
- 7) If $K_i \equiv \text{open } b$, then $a[\text{open } b.P_i] \mapsto a[\text{open } b].a[_ | P_i]$;
- 8) If $K_i \equiv \overline{\text{open}} b$, then $a[\overline{\text{open}} b.P_i] \mapsto a[\overline{\text{open}} b].\langle P_i \rangle$;
- 9) Otherwise, $a[K_i.P_i] \mapsto \mathbf{0}$.

The case of $K_i \equiv [\text{out } b]$ is worth mentioning, since we write a form of $P_i \equiv (\tilde{x})(\langle P'_i \rangle | P''_i)$ next to the prefix $[\text{out } b]$, presumptively. This is because we can easily verify that it is the solely form next to a prefix like $[\text{out } b]$, provided that $K_i.P_i$ is already in normal form.

- Faced with the last case of parallel composition $P_1 | P_2$, our procedure relies on the law of expansion, rewriting the left hand side to the right hand side:

Theorem 3. *Suppose both $P \equiv \sum K_m.P_m$ and $Q \equiv \sum K_n.P_n$ are in normal form, and $\tilde{y} \cap \tilde{z} = \emptyset$, then*

$$\begin{aligned} P | Q &= \sum K_m.(P_m | \sum K_n.Q_n) \\ &+ \sum K_n.(\sum K_m.P_m | Q_n) \\ &+ \tau.(\tilde{y})(\tilde{z})(b[P'_j | P'_k] | P_j | P_k) \\ &\quad \left(\text{for each } \begin{cases} a[\text{in } b].(\tilde{y})(\langle P'_j \rangle | P_j) \\ b[\overline{\text{in}} a].(\tilde{z})(b[_ | P'_k] | P_k) \end{cases} \right) \\ &+ \tau.(\tilde{y})(\tilde{z})(P_j | P_k) \\ &\quad \left(\text{for each } \begin{cases} a[[\text{out } b]].(\tilde{y})P_j \\ b[\overline{\text{out}} a].(\tilde{z})P_k \end{cases} \right) \\ &+ \tau.(\tilde{y})(\tilde{z})(b[P'_j | P'_k] | P_j | P_k) \\ &\quad \left(\text{for each } \begin{cases} a[\text{open } b].(\tilde{y})(\langle P'_j \rangle | P_j) \\ b[\overline{\text{open}} a].(\tilde{z})(b[_ | P'_k] | P_k) \end{cases} \right) \end{aligned}$$

It is easy to see that the right hand side of the above equation is of the form $\sum K_i.P_i$, and each $K_i.P_i$ falls into the three categories of a normal form's component parts

$$K'.(\tilde{x})P', K'.(\tilde{x})(\langle P'' \rangle | P'), K'.(\tilde{x})(a[_ | P''] | P')$$

Although P' or P'' may not be in normal form, we can process them recursively to rewrite all of them into normal form due to the fact

$$\max\{d(P'), d(P'')\} \leq d(P | Q) - 1$$

Since we are dealing with finite fair ambient processes, by an easy structural induction, this procedure will surely terminate. When it terminates, a strong bisimilar normal form extended fair ambient is output, which is to be demonstrated. ■

IV. AXIOMS FOR STRONG BISIMILARITY

As the first step, we investigate the axiom system of strong bisimilarity \sim , since it is often considered as a base for axiomatizing other equivalence relations.

Table I lists the axioms of strong bisimilarity on finite fair ambient (**ASI**), which are divided into four groups:

- **[E-axioms]** stand for an equivalence relation;
- **[C-axioms]** are here for deriving congruence, because strong bisimilarity is congruent for all operators;
- **[L-axioms]** deal with the restriction operator, which is also known as the localization operator;
- **[S-axioms]** handle the guarded choice operator, which is called summation operator in the literature.

The main theorem in this subsection is the following theorem.

Theorem 4 (Soundness and Completeness). $P \sim Q$ iff $ASI \vdash P = Q$.

Firstly, we establish the soundness part of it.

Proof of Soundness: The only group of axioms merits attention is the group of C-axioms, which are used to derive congruent property of strong bisimilarity.

Among them, **C1**, **C3** and **C5** have been proved in [9] as a result of the congruent of \sim in FA. **C4** is always true since there is no labeled transition system rule for a concretion in the operational semantics of extended fair ambient. The remainder, **C2**, is valid because this time the equal sign “=” refers to strong bisimilarity “ \sim ”, where action τ does not have the preemptive power over summation as in the weak version. When it refers to weak bisimilarity “ \approx ”, discussed in the next subsection, **C2** remains valid for the processes rewritten from an ordinary FA process, details postponed till next subsection.

Cases for the other three groups of axioms are either obvious or can be easily established. ■

As mentioned earlier, concretions and contexts are intermediate forms, so bisimulation relations are not defined directly on them. Instead, we use a universally quantifier over all possible final forms of the interaction, e.g.

A symmetric relation \mathcal{R} on processes is a strong bisimulation if the following hold whenever $P \mathcal{R} Q$:

- ...
5. If $P \xrightarrow{a[\text{in } b]} (\tilde{m})\langle A \rangle P'$, then for every process O satisfying $\tilde{m} \cap fn(O) = \emptyset$, there exist some \tilde{n} , B , Q' such that $Q \xrightarrow{a[\text{in } b]} (\tilde{n})\langle B \rangle Q'$ and $(\tilde{n})(Q' | b[B | O]) \mathcal{R} (\tilde{m})(P' | b[A | O])$.
- ...

7. If $P \xrightarrow{a[\text{in } b]} C[_]$, then for every N satisfying $fn(N) \cap bn(C[_]) = \emptyset$, some $C'[_]$ exists such that $Q \xrightarrow{a[\text{in } b]} C'[_]$ and $C'[b[N]] \mathcal{R} C[b[N]]$.
- ...

In axiomatizing, concretions and contexts are terms directly manipulated by equations. It is essential that the equal sign “=” actually captures strong bisimilarity, and next lemma gives an account of it.

Lemma 1 (Abstraction). *Suppose c is fresh, then $\forall R_1, R_2$:*

$$(\tilde{x})(a[P_1 | R_1] | P_2 | R_2) \approx (\tilde{x})(a[Q_1 | R_1] | Q_2 | R_2)$$

and

$$(\tilde{x})(a[P_1 | \text{open } c] | P_2) \approx (\tilde{x})(a[Q_1 | \text{open } c] | Q_2)$$

are equivalent.

Proof: “ \implies ”: This direction is clear.
 “ \impliedby ”: Let

$$C_1[_] = a[_ | P_1] | P_2$$

$$C_2[_] = a[_ | Q_1] | Q_2$$

and we define

$$\mathcal{R} = \{(C_1[R], C_2[R]) \mid C_1[\text{open } c] \approx C_2[\text{open } c], \\ R \text{ is an arbitrary process}\} \cup \approx$$

Then we show \mathcal{R} to be a bisimulation up-to \sim . Suppose $C_1[R] \xrightarrow{K} P'$,

- K is on R , that is, $R \xrightarrow{K} R'$ and $C_1[R] \xrightarrow{K} C_1[R']$. Thus we have

$$C_1[\text{open } c] \xrightarrow{a[\text{open } c]} C_1[_]$$

and since $C_1[\text{open } c] \approx C_2[\text{open } c]$, this is simulated by

$$C_2[\text{open } c] \xrightarrow{a[\text{open } c]} C_2[_]$$

with $C_1[R'] \approx C_2[R']$. So

$$C_2[R] \xrightarrow{K} C_2[R'] \approx C_1[R']$$

- K is on $C_1[_]$, that is, $C_1[R] \xrightarrow{K} C_1'[R]$. Then

$$C_1[\text{open } c] \xrightarrow{K} C_1'[\text{open } c]$$

and this is simulated by

$$C_2[\text{open } c] \xrightarrow{K} C_2'[\text{open } c] \approx C_1'[\text{open } c]$$

since $C_1[\text{open } c] \approx C_2[\text{open } c]$. So the simulation step is

$$C_2[R] \xrightarrow{K} C_2'[R] \approx C_1'[R]$$

- K is τ , and it is caused by a communication between $C_1[_]$ and R . Suppose $C_1[_] \xrightarrow{\lambda} C_1'[_]$ and $R \xrightarrow{\lambda} R'$, and let $E \equiv c[\overline{\text{open } a}.\bar{\lambda}.R']$. then

$$C_1[\text{open } c] | E \equiv a[\text{open } c | P_1] | P_2 | c[\overline{\text{open } a}.\bar{\lambda}.R']$$

$$\xrightarrow{\tau} a[\bar{\lambda}.R' | P_1] | P_2$$

$$\xrightarrow{\tau} C_1'[R']$$

E1	$P = P$	
E2	$P = Q$	if $Q = P$
E3	$P = R$	if $P = Q$ and $Q = R$
C1	$K.P = K.Q$	if $P = Q$
C2	$P + R = Q + R$	if $P = Q$
C3	$(a)P = (a)Q$	if $P = Q$
C4	$P R = Q R$	if $P = Q$
C5	$\langle P \rangle = \langle Q \rangle$	if $P = Q$
L1	$(a)\mathbf{0} = \mathbf{0}$	
L2	$(a)(b)P = (b)(a)P$	
L3	$(a)(P + Q) = (a)P + (a)Q$	
L4	$(a)K.P = K.(a)P$	if $a \notin n(K)$
L5	$(a)K.P = \mathbf{0}$	if $a \in n(K)$
L6	$(a)(P Q) = (a)P Q$	if $a \notin fn(Q)$
S1	$P + \mathbf{0} = P$	
S2	$P + Q = Q + P$	
S3	$(P + Q) + R = P + (Q + R)$	
S4	$P + P = P$	

TABLE I
THE AXIOM SYSTEM OF STRONG BISIMILARITY: **AS1**

must be simulated by

$$\begin{aligned}
C_2[\text{open } c] | E &\equiv a[\text{open } c | Q_1] | Q_2 | c[\overline{\text{open}} a.\bar{\lambda}.R'] \\
&\stackrel{\tau}{\Rightarrow} a[\bar{\lambda}.R' | Q_1] | Q_2 \\
&\stackrel{\tau}{\Rightarrow} C'_2[R'] \\
&\approx C'_1[R']
\end{aligned}$$

Thus we have

$$C_2[R] \stackrel{\tau}{\Rightarrow} C'_2[R'] \approx C'_1[R']$$

Lemma 2 (Separation). *For each process R with $fn(R) \not\subseteq \tilde{x} \cup \tilde{y}$,*

$$(\tilde{x})(a[R | P_1] | P_2) \sim (\tilde{y})(b[R | Q_1] | Q_2)$$

if and only if there exists an injective substitution σ renaming \tilde{y} to \tilde{x} (i.e. $\tilde{y}\sigma = \tilde{x}$) such that

$$a = b\sigma \wedge P_1 \sim Q_1\sigma \wedge P_2 \sim Q_2\sigma$$

Proof: “ \Leftarrow ”: This direction is somewhat straightforward. Since $P_1 \sim Q_1\sigma$ and $R \equiv R\sigma$, we have $R | P_1 \sim R\sigma | Q_1\sigma$. Then by $P_2 \sim Q_2\sigma$ and $a = b\sigma$, the following holds,

$$a[R | P_1] | P_2 \sim b\sigma[R\sigma | Q_1\sigma] | Q_2\sigma$$

Again, according to the congruence property of strong bisimilarity,

$$(\tilde{x})(a[R | P_1] | P_2) \sim (\tilde{x})(b\sigma[R\sigma | Q_1\sigma] | Q_2\sigma)$$

Let σ^{-1} be the inverse function of σ , α -convertibility says

$$\begin{aligned}
(\tilde{x})(a[R | P_1] | P_2) &\sim (\tilde{x})(b\sigma[R\sigma | Q_1\sigma] | Q_2\sigma) \\
&\sim (\tilde{x}\sigma^{-1})(b\sigma[R\sigma | Q_1\sigma] | Q_2\sigma)\sigma^{-1} \\
&\sim (\tilde{y})(b[R | Q_1] | Q_2)
\end{aligned}$$

“ \Rightarrow ”: By contradiction, suppose $P_1 \sim Q_1 \wedge P_2 \sim Q_2 \wedge a = b$ does not hold, then there are three cases:

- 1) $a \neq b$. It is easy by choosing $R = \text{open } c.\mathbf{0}$ where c is fresh;

- 2) $P_1 \approx Q_1$. Suppose $P_1 \xrightarrow{\lambda}$ but $Q_1 \not\xrightarrow{\lambda}$, then let $R = \bar{\lambda}.\mathbf{0}$;

- 3) $P_2 \approx Q_2$. Just choose $R = \mathbf{0}$.

To establish the completeness part, we need to transform each FA process into normal form via the rewritten algorithm.

Proof of Completeness: Assuming $P \sim Q$ and both

$$\begin{aligned}
P &\equiv \sum K_i.(\tilde{x})P_i + \sum K_j.(\tilde{y})(\langle P'_j \rangle | P_j) \\
&\quad + \sum K_k.(\tilde{z})(a_k[_- | P'_k] | P_k) \\
Q &\equiv \sum K'_i.(\tilde{x}')Q_i + \sum K'_j.(\tilde{y}')(\langle Q'_j \rangle | Q_j) \\
&\quad + \sum K'_k.(\tilde{z}')(a'_k[_- | Q'_k] | Q_k)
\end{aligned}$$

are in normal form, we prove the result by induction on the maximum depth of P and Q .

If the maximum depth of P and Q is 0, then P and Q are both $\mathbf{0}$ (since they are in normal form), and **AS1** $\vdash \mathbf{0} = \mathbf{0}$.

Otherwise, for each summand of P ,

- 1) $K_i.(\tilde{x})P_i$ ($K_i \in \{\tau, \kappa, a[\text{out } b], a[\overline{\text{out}} b] \mid a, b \in \mathcal{N}\}$). Then $P \xrightarrow{K_i} (\tilde{x})P_i$ so since $P \sim Q$ there must be some Q'_i such that $Q \xrightarrow{K_i} Q'_i \sim (\tilde{x})P_i$. But Q is in normal form, so $Q'_i \equiv (\tilde{x}')Q_i$ and $K_i.(\tilde{x})Q_i$ is a summand of Q . By induction we have

$$\mathbf{AS1} \vdash (\tilde{x})P_i = (\tilde{x}')Q_i$$

which implies

$$\mathbf{AS1} \vdash K_i.(\tilde{x})P_i = K_i.(\tilde{x}')Q_i$$

- 2) $K_j.(\tilde{y})(\langle P'_j \rangle | P_j)$ ($K_j \equiv [\text{out } a]$). Then $P \xrightarrow{K_j} (\tilde{y})(\langle P'_j \rangle | P_j)$, so since $P \sim Q$ there must be \tilde{y}' , Q_j and Q'_j such that $Q \xrightarrow{K_j} (\tilde{y}')(\langle Q'_j \rangle | Q_j)$ and for every name d and every process O ,

$$(\tilde{y}')(Q_j | d[Q'_j | O]) \sim (\tilde{y})(P_j | d[P'_j | O])$$

By lemma, we have

$$Q_j \sim P_j \wedge Q'_j \sim P'_j \wedge \tilde{y}' = \tilde{y}$$

then by induction,

$$\mathbf{AS1} \vdash Q_j = P_j$$

$$\mathbf{AS1} \vdash Q'_j = P'_j$$

Finally, according to C-axioms,

$$\mathbf{AS1} \vdash \langle Q'_j \rangle = \langle P'_j \rangle$$

$$\vdash \langle Q'_j \rangle | Q_j = \langle P'_j \rangle | P_j$$

$$\vdash (\tilde{y}')(\langle Q'_j \rangle | Q_j) = (\tilde{y})(\langle P'_j \rangle | P_j)$$

$$\vdash K_j.(\tilde{y}')(\langle Q'_j \rangle | Q_j) = K_j.(\tilde{y})(\langle P'_j \rangle | P_j)$$

- 3) $K_j.(\tilde{y})(\langle P'_j \rangle | P_j)$ ($K_j \in \{a[\text{in } b], a[\overline{\text{open}} b]\}$). This case is similar to $K_j \equiv [\text{out } b]$, except that d equals to b and does not range over N .
- 4) $K_k.(\tilde{z})(a_k[_\perp | P'_k] | P_k)$ ($K_k \equiv a[\overline{\text{in}} b]$). Then $P \xrightarrow{K_k} (\tilde{z})(a_k[_\perp | P'_k] | P_k)$, so since $P \sim Q$ there must be $C'[_\perp]$ such that $Q \xrightarrow{K_k} C'[_\perp]$ and for every process N ,

$$C'[b[N]] \sim (\tilde{z})(a_k[b[N] | P'_k] | P_k)$$

But Q is in normal form, so

$$C'[_\perp] \equiv (\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k)$$

By lemma, we have

$$Q_k \sim P_k \wedge Q'_k \sim P'_k \wedge \tilde{z}' = \tilde{z} \wedge a'_k = a_k$$

then by induction,

$$\mathbf{AS1} \vdash Q_k = P_k$$

$$\mathbf{AS1} \vdash Q'_k = P'_k$$

Finally, according to C-axioms,

$$\mathbf{AS1} \vdash _\perp | Q'_k = _\perp | P'_k$$

$$\vdash a'_k[_\perp | Q'_k] = a_k[_\perp | P'_k]$$

$$\vdash a'_k[_\perp | Q'_k] | Q_k = a_k[_\perp | P'_k] | P_k$$

$$\vdash (\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k) = (\tilde{z})(a_k[_\perp | P'_k] | P_k)$$

$$\vdash K_k.(\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k)$$

$$= K_k.(\tilde{z})(a_k[_\perp | P'_k] | P_k)$$

- 5) $K_k.(\tilde{z})(a_k[_\perp | P'_k] | P_k)$ ($K_k \equiv a[\text{open } b]$). Then $P \xrightarrow{K_k} (\tilde{z})(a_k[_\perp | P'_k] | P_k)$, so since $P \sim Q$ for every N satisfying $fn(N) \cap bn(C[_\perp]) = \emptyset$, there must be $C'[_\perp]$ such that $Q \xrightarrow{K_k} C'[_\perp]$ and

$$C'[N] \sim (\tilde{z})(a_k[N | P'_k] | P_k)$$

But Q is in normal form, so

$$C'[_\perp] \equiv (\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k)$$

Choose N to be $\text{open } c$ for a fresh name c , then

$$(\tilde{z})(a_k[\text{open } c | P'_k] | P_k) \sim (\tilde{z}')(a'_k[\text{open } c | Q'_k] | Q_k)$$

We have

$$\mathbf{AS1} \vdash _\perp | Q'_k = _\perp | P'_k$$

$$\vdash a'_k[_\perp | Q'_k] = a_k[_\perp | P'_k]$$

$$\vdash a'_k[_\perp | Q'_k] | Q_k = a_k[_\perp | P'_k] | P_k$$

$$\vdash (\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k) = (\tilde{z})(a_k[_\perp | P'_k] | P_k)$$

$$\vdash K_k.(\tilde{z}')(a'_k[_\perp | Q'_k] | Q_k)$$

$$= K_k.(\tilde{z})(a_k[_\perp | P'_k] | P_k)$$

So every summand of P can be proved equal to a summand of Q . Similarly, every summand of Q can be proved equal to a summand of P . It follows that $\mathbf{AS1} \vdash P = Q$, by using S-axioms to eliminate duplicate summands and reorder them as necessary. ■

V. CHECKING BISIMILARITY

The axiom system provides us with an approach to determinate whether two finite fair ambient processes are bisimilar. Firstly, we should translate each finite process into normal form. Then we check if they are provably equal in our axiom system.

Next we provide many examples to show the actual situations in checking bisimilarity.

Example 6 (Restriction).

$$\begin{aligned} 3(a)a[\text{out } b.M] &= (a)[\text{out } b].\langle a[M] \rangle \\ &= [\text{out } b].(a)\langle a[M] \rangle \neq \mathbf{0} \\ (a)c[\text{out } a.M] &= (a)[\text{out } a].\langle c[M] \rangle \\ &= \mathbf{0} \end{aligned}$$

Example 7 (How to compare two ambient via their normal forms).

$$\begin{aligned} a[\overline{\text{in}} b | c[\text{open } b]] &= a[\overline{\text{in}} b | c[\text{open } b].c[_\perp]] \\ &= a[\overline{\text{in}} b.c[\text{open } b].c[_\perp] \\ &\quad + c[\text{open } b].(c[_\perp] | \overline{\text{in}} b)] \\ &= a[\overline{\text{in}} b].a[_\perp | c[\text{open } b].c[_\perp]] \\ a[\overline{\text{in}} b] &= a[\overline{\text{in}} b].a[_\perp] \\ a[\overline{\text{in}} b | c[\text{open } b]] &\neq a[\overline{\text{in}} b] \end{aligned}$$

Example 8. The following examples are taken from [9]. Here we translate each process into normal form, then we can find that the checking results are just what the definition of bisimilarity tells us.

- $a[\mathbf{0}] \approx c[a[\mathbf{0}] | b[\mathbf{0}]]$:

$$a[\mathbf{0}] = \mathbf{0}$$

$$c[a[\mathbf{0}] | b[\mathbf{0}]] = \mathbf{0}$$

- $(a)a[\text{out } b.P] \not\approx \mathbf{0}$:

$$\begin{aligned} (a)a[\text{out } b.P] &= [\text{out } b].(a)\langle a[P] \rangle \\ &\neq \mathbf{0} \end{aligned}$$

- $(a)a[n[N] \mid \text{in } b.M] \approx \mathbf{0}$, suppose $n[N] = \sum K_i.P_i + \sum \tau.P_\tau$:

$$\begin{aligned}
& (a)a[n[N] \mid \text{in } b.M] \\
&= (a)a[(\sum K_i.P_i + \sum \tau.P_\tau) \mid \text{in } b.M] \\
&= (a)a[\sum K_i.(P_i \mid \text{in } b.M) \\
&\quad + \sum \tau.(P_\tau \mid \text{in } b.M) + \text{in } b.(n[N] \mid M)] \\
&= \sum \tau.(a)(a[P_\tau \mid \text{in } b.M]) \\
&= \sum \tau.(\sum \tau.(\dots \sum \tau.\mathbf{0})) \\
&= \mathbf{0}
\end{aligned}$$

- $(a)a[n[N] \mid \text{out } b.M] \approx (a)a[\text{out } b.(n[N] \mid M)]$, as supposed above:

$$\begin{aligned}
& (a)a[n[N] \mid \text{out } b.M] \\
&= (a)a[\sum \tau.(P_\tau \mid \text{out } b.M) + \text{out } b.(n[N] \mid M)] \\
&= \sum \tau.(a)(a[P_\tau \mid \text{out } b.M]) \\
&\quad + [\text{out } b].(a)\langle a[n[N] \mid M] \rangle \\
&= (a)a[\text{out } b.(n[N] \mid M)] \\
&= [\text{out } b].(a)\langle a[n[N] \mid M] \rangle
\end{aligned}$$

- $(c)a[b[\text{out } c.A]] \approx \mathbf{0}$ for fresh c :

$$\begin{aligned}
(c)a[b[\text{out } c.A]] &= (c)a[[\text{out } c].\langle b[A] \rangle] \\
&= (c)a[[\text{out } c]].b[A] \\
&= \mathbf{0}
\end{aligned}$$

- $(d)b[\text{out } a.\text{in } d] \approx c[\text{out } a]$:

$$\begin{aligned}
(d)b[\text{out } a.\text{in } d] &= (d)[\text{out } a].\langle b[\text{in } d] \rangle \\
&= [\text{out } a].(d)\langle b[\text{in } d].\langle \mathbf{0} \rangle \rangle \\
&= [\text{out } a].\langle (d)b[\text{in } d] \rangle \\
&= [\text{out } a].\langle \mathbf{0} \rangle \\
&= [\text{out } a] \\
c[\text{out } a] &= [\text{out } a].\langle c[\mathbf{0}] \rangle \\
&= [\text{out } a]
\end{aligned}$$

Let $n[N] = \sum K_i.P_i$, the derivation in the above example is:

$$\begin{aligned}
(a)a[n[N] \mid \text{out } b.M] &= (a)a[\sum K_i.(P_i \mid \text{out } b.M)] \\
&\quad + b.(M \mid \sum K_i.P_i) \\
&= \tau.a[P_i \mid \text{out } b.M] \\
&\quad + a[[\text{out } c]].(Q \mid a[Q' \mid \text{out } b.M]) \\
&\quad + [\text{out } b].\langle a[M \mid \sum K_i.P_i] \rangle
\end{aligned}$$

VI. CONCLUSION

The use of extended Fair Ambient normal forms are crucial to the establishment of axiom system. Because FA is inherently higher order, avoiding higher order labels in the operational semantics are necessary. Here concretions and contexts are introduced into extended Fair Ambient calculus. In addition to concretions and contexts,

unguarded choice also should be used to make axiomatization smooth. The ambient structure are transformed into prefixes. These points are original key points in the structure of normal form.

As for the future work, we are currently examining the axiom system for weak bisimilarity and open bisimilarity. These axiom systems will further the study of ambient calculi, and to our best knowledge they remain untouched. We are also working on the automatical checking tools based on our approach. In our opinion, these breakthroughs can give us deeper insight into the ambient calculi.

ACKNOWLEDGEMENTS.

The author would like to express his gratitude to Prof. Yuxi Fu for his inspiration, guidance and many fruitful discussions on this topic. The author is indebted to members of Basics Lab for their proofreading of this paper and many useful suggestions on improvements.

The author also wish to thank anonymous referees for their constructive comments.

REFERENCES

- [1] L. Cardelli and A. Gordon, "Types for mobile ambients," in *Proceedings of POPL'99*, 1999.
- [2] —, "Anytime, anywhere: Modal logics for mobile ambients," in *Proceedings of POPL'00*, 2000.
- [3] —, "Mobile ambients," *Theoretical Computer Science*, vol. 240, no. 1, pp. 177–213, 2000.
- [4] L. Cardelli, "Abstraction for mobile computation," *Lecture Notes in Computer Science*, vol. 1603, 1999.
- [5] F. Levi and D. Sangiorgi, "Controlling interference in ambients," in *Proceedings of the 27th Symposium on Principles of Programming Languages (POPL'00)*. ACM Press, 2000, pp. 352–364.
- [6] M. Merro and M. Hennessy, "Bisimulation congruences in safe ambients," in *Proceedings of POPL'02*, 2002, pp. 71–80.
- [7] D. Teller, P. Zimmer, and D. Hirschcoff, "Using ambients to control resources," in *Proceedings of CONCUR'02*, ser. Lecture Notes in Computer Science, vol. 2421, 2002, pp. 288–303.
- [8] X. Guan, Y. Yang, and J. You, "Typing evolving ambients," *Information Processing Letters*, vol. 80, pp. 265–270, 2001.
- [9] Y. Fu, "Fair ambients," *Acta Informatica*, vol. 43, no. 8, pp. 535–594, 2007.
- [10] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes, part i," *Information and Computation*, vol. 100, no. 1, pp. 1–40, 1992.
- [11] —, "A calculus of mobile processes, part ii," *Information and Computation*, vol. 100, no. 1, pp. 41–77, 1992.

Han Zhu was born in 1981, Shanghai, China.

He got his Bachelor of Science from Shanghai Jiao Tong University in 2003. He now is a candidate of Ph.D in Department of CS in Shanghai Jiao Tong University, supervised by Prof. Yuxi Fu. His main research interests include process algebra, security protocols and formal methods.